Bem-Vindo ao curso de lógica de programação para Games e Aplicativos.

Eu sou o **Tito Petri** e este é o meu Mascotinho, o **Felpudo**.

Veja o que aprenderemos nesta aula:

- Lógica de Programação e Algoritmos.
- Programação no Scratch.
- Introdução às Linguagens Java, C-Sharp, Swift e Python.

Para quem serve este treinamento:

- Iniciantes em Programação
- Alunos de T.I. ou Ciência da Computação
- Designers e Animadores que desejam iniciar na Programação de Jogos e Apps
- Alunos com dificuldade em aprender Lógica e Algoritmos

Variáveis e Tipos

- O que são Variáveis?
- Tipos de Variáveis (boolean, int, float, double, string)
- Declaração de Variáveis (Scratch, Java, C#, Swift, Python)
- Acessando e Utilizando Variáveis e Operador **=**

Variáveis no Scratch

	Declaração e Uso de Variáveis no Scratch

Acessando a categoria *Variáveis* podemos criar uma nova variável e modificá-la pelos blocos ***mude_para*** ou ***adicione_a***.

Quando criamos uma nova variável, podemos escolher se ela será utilizada apenas no ***Ator atual*** ou em **todos os *Atores*** do projeto.

Também há a possibilidade de guardar a variável na nuvem, para por exemplo armazenar um dado de recorde entre os jogadores.

Variáveis em **Java**

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {

        // Isto é só um comentário em Java! =)
        System.out.println("Olá eu sou o Felpudo!");

        //Declaração de Variáveis:
        String nome = "Tito Petri";
        boolean casado = true;
        int idade = 35;
        float altura = 1.82f;
        double peso = 80.5;

        // Acessando uma variável:
        idade = 20;
        altura = 1.50f;
        peso = 50;
        System.out.println("Nome: " + nome + "\nPeso: " + peso + "\nAltura: "  + altura);
    }
}
```

Variáveis em **C-Sharp C#**

```
public class Hello{
    public static void Main(){
        // Isto é só um comentário em C-Sharp C#! =)
        System.Console.WriteLine("Olá eu sou o Felpudo!");

        //Declaração de Variáveis:
        string nome = "Tito Petri";
        bool casado = true;
        int idade = 35;
        float altura = 1.82f;
        double peso = 80.5;

        // Acessando uma variável:
        idade = 20;
        altura = 1.50f;
        peso = 50;
        System.Console.WriteLine("Nome: " + nome + "\nPeso: " + peso + "\nAltura: "  + altura);

    }
}
```

Variáveis em **Swift**

```
// Isto é só um comentário em Swift! =)
print("Olá eu sou o Felpudo!")

//Declaração de Variáveis:
let nome:String = "Tito Petri"
let casado:Bool = true
var idade:Int = 35
var altura = 1.82
var peso = 80.5

// Acessando uma variável:
idade = 20
altura = 1.50
peso = 50

print("Nome: \(nome)\nIdade: \(idade)\nAltura: \(altura)\nPeso: \(peso)")
```

Variáveis em **Python**

```
# coding: utf-8
# Isto é só um comentário em Python! =)
print("Olá eu sou o Felpudo!")

# Declaração de Variáveis:
nome = "Tito Petri"
casado = True
idade = 35
altura = 1.82
peso = 80.5

# Acessando uma variável:
idade = 20
altura = 1.50
peso = 50

print("Nome: "+nome+"\nIdade: "+str(idade)+"\nAltura: "+str(altura)+"\nPeso: "+str(peso)+"\nCasado: 
"+str(casado))
```

Palavras e Termos Técnicos

Algoritmo é a resolução de um problema (receita).

Sintaxe é a forma como você escreve esta receita (comandos de cada linguagem).

Script é um texto com uma série de instruções executadas por um programa de computador.

Escopo é o contexto delimitante aos quais variáveis e funções estão associados.

Lógica é uma palavra que define a ciência do raciocínio.

Case Sensitive "sensível à caixa das letras" ou "sensível a maiúsculas e minúsculas".

Camel Case escrita de palavras ou frases compostas, onde cada palavra é iniciada com maiúsculas e unidas sem espaços.

Exemplo: **olaeusouofelpudo** / **OlaEuSouOFelpudo**

Operadores Aritméticos e Operadores Unários

- Operações Básicas da Matemática
- Adição, Subtração, Multiplicação e Divisão
- Representados pelos caracteres **+ - * /**
- O Módulo retorna o resto da divisão, representado pelo caracter **%**
- **Operadores unários** são operadores aritméticos que desempenham uma ação em um único operando.

```java
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {

        int meuNumero = 50;

        System.out.println(meuNumero+100);
        System.out.println(meuNumero-100);
        System.out.println(meuNumero*100);
        System.out.println(meuNumero/100);
        System.out.println(7%3);

        // meuNumero = meuNumero + 1;
        meuNumero *= 2;
        System.out.println(meuNumero);
    }
}
```

```csharp
public class Hello{
    public static void Main(){
        int meuNumero = 50;

        System.Console.WriteLine(meuNumero+100);
        System.Console.WriteLine(meuNumero-100);
        System.Console.WriteLine(meuNumero*100);
        System.Console.WriteLine(meuNumero/100);
        System.Console.WriteLine(7%3);

        // meuNumero = meuNumero + 1;
        meuNumero *= 2;
        System.Console.WriteLine(meuNumero);
    }
}
```

```
var meuNumero = 50;

print(meuNumero+100);
print(meuNumero-100);
print(meuNumero*100);
print(meuNumero/100);
print(7%3);

// meuNumero = meuNumero + 1;
```

```
meuNumero *= 2;
print(meuNumero);
```

```
# coding: utf-8
# Your code here!

meuNumero = 50;

print(meuNumero+100);
print(meuNumero-100);
print(meuNumero*100);
print(meuNumero/100);
print(7%3);

# meuNumero = meuNumero + 1;
meuNumero *= 2;
print(meuNumero);
```

Booleanos

True False

Operadores Lógicos

São a base para a construção de sistemas digitais e da Lógica proposicional.

- **NOT** - Representado pelo caracter **!**
- **AND** - Representado pelo caracter **&&**
- **OR** - Representado pelo caracter **||**

Operadores de Comparação

Compara dois valores e retorna um *boolean*

- **Igual** - Representado pelo caracter **==**
- **Diferente** - Representado pelo caracter **!=**

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        //  Booleanos
         boolean estado = false;
         boolean ligado = true;

        //  Operadores Lógicos
        // NOT
        System.out.println(!estado);
        // OR
        System.out.println(estado || ligado);
        // AND
        System.out.println(estado && ligado);

        // Operadores de Comparação
        //  > < >= <= == !=
        System.out.println(10 > 5);
        System.out.println(10 < 5);
        System.out.println(10 >= 10);
        System.out.println(10 != 5);
        System.out.println(10 == 1);
```

```
    }
}
```

```
public class Hello{
    public static void Main(){
        //  Booleanos
         bool estado = false;
         bool ligado = true;

        //  Operadores Lógicos
        // NOT
        System.Console.WriteLine(!estado);
        // OR
        System.Console.WriteLine(estado || ligado);
        // AND
        System.Console.WriteLine(estado && ligado);

        // Operadores de Comparação
        //  > < >= <= == !=
        System.Console.WriteLine(10 > 5);
        System.Console.WriteLine(10 < 5);
        System.Console.WriteLine(10 >= 10);
        System.Console.WriteLine(10 != 5);
        System.Console.WriteLine(10 == 1);
    }
}
```

```
var estado = false;
var ligado = true;

//  Operadores Lógicos
// NOT
print(!estado);
// OR
print(estado || ligado);
// AND
print(estado && ligado);

// Operadores de Comparação
//  > < >= <= == !=
print(10 > 5);
print(10 < 5);
print(10 >= 10);
print(10 != 5);
print(10 == 1);
```

```
# coding: utf-8
# Your code here!
estado = False
ligado = True

# Operadores Lógicos
# NOT
print(not estado)
# OR
print(estado or ligado)
# AND
print(estado and ligado)

# Operadores de Comparação
#  > < >= <= == !=
print(10 > 5)
print(10 < 5)
print(10 >= 10)
print(10 != 5)
print(10 == 1)
```

Listas ou Arrays

Estrutura que armazena uma coleção de variáveis do mesmo tipo, que podem ser acessadas através de um índice.

O primeiro índice é sempre o 0 Zero.

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        String[] minhaLista = {"Felpudo", "Fofura", "Lesmo", "Bugado", "Uruca"};

        minhaLista[0] = "Tito Petri";
        System.out.println(minhaLista[0]);

    }
}
```

```
public class Hello{
    public static void Main(){

        string[] minhaLista = {"Felpudo", "Fofura", "Lesmo", "Bugado", "Uruca"};
        minhaLista[0] = "Tito Petri";
        System.Console.WriteLine(minhaLista[0]);

    }
}
```

```
var minhaLista:[String] = ["Felpudo", "Fofura", "Lesmo", "Bugado", "Uruca"];
minhaLista[0] = "Tito Petri";
print(minhaLista[0]);
```

```
# coding: utf-8
# Your code here!

minhaLista = ["Felpudo", "Fofura", "Lesmo", "Bugado", "Uruca"];
minhaLista[0] = "Tito Petri";
print(minhaLista[0]);
```

If Else

A condição Se / Senão é a tomada de decisão pela aplicação digital.
É a hora que o programa escolhe se vai pra direita ou pra esquerda.

Uma estrutura de decisão examina uma ou mais **condições** e decide quais instruções serão executadas dependendo se a **condição** foi ou não foi.

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {

        int idade = 18;
```

``
        if(idade<18){
            System.out.println("Não pode dirigir.");
        }else if (idade >= 60){
            System.out.println("Deve renovar a carta.");
        }else{
            System.out.println("Pode dirigir.");
        }
    }
}
``

```
        if(idade<18){
            System.out.println("Não pode dirigir.");
        }else if (idade >= 60){
            System.out.println("Deve renovar a carta.");
        }else{
            System.out.println("Pode dirigir.");
        }
    }
}
```

```
public class Hello{
    public static void Main(){

        int idade = 18;

        if(idade<18){
            System.Console.WriteLine("Não pode dirigir.");
        }else if (idade >= 60){
            System.Console.WriteLine("Deve renovar a carta.");
        }else{
            System.Console.WriteLine("Pode dirigir.");
        }
    }
}
```

```
var idade = 18;

if(idade<18){
    print("Não pode dirigir.");
}else if (idade >= 60){
    print("Deve renovar a carta.");
}else{
    print("Pode dirigir.");
}
```

```
# coding: utf-8
# Your code here!

idade = 18

if (idade < 18):
    print("Não Pode Dirigir")
elif (idade > 60):
    print("Deve renovar a carta")
else:
    print("Pode Dirigir")
```

For

O Loop ou Laço de Repetição For / Para
Executa uma determinada instrução repetidamente por determinadas vezes
Presença de um contador, normalmente apelidado de **i**.

Argumentos necessários para a criação de um Loop For:

- **Declaração** do Contador
- **Comparação**
- **Incremento**

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {

        for(int i=0;i<10;i++){
            System.out.println("Valor do Contador: " + i);
        }
    }
}
```

```
public class Hello{
    public static void Main(){
        for(int i=0;i<10;i++){
            System.Console.WriteLine("Valor do Contador: " + i);
        }
    }
}
```

```
for i in 0..<10{
    print("Valor do Contador: \(i)");
}
```

```
# coding: utf-8
# Your code here!

for i in range(10):
    print("Valor do Contador: " + str(i))
```

Métodos Funções Procedimentos

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        // Chamada ou Uso dos Métodos
        digaOla();

        somar(10,5);
        subtrair(100,30);
        multiplicar(10,3);
        dividir(30,5);

        float resultado = dividir(100,300);
        System.out.println(resultado);
    }

    // Procedimento
    static void digaOla(){
        System.out.println("Olá, eu sou o Felpudo!");
    }

    // Procedimento com passagem de Argumentos
    static void somar(int a, int b){
        System.out.println(a-b);
    }

    // Funções com Retorno de Valor e passagem de Argumentos
    static float subtrair(float minuendo, float subtraendo){
        float diferenca = minuendo-subtraendo;
        return diferenca;
    }

    static float multiplicar(float a, float b){
        return a*b;
    }
    static float dividir(float a, float b){
        return a/b;
    }
}
```

Classes e Hierarquias

```
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {

        Animal animal = new Animal();
        System.out.println(animal.nome);
        animal.falar();

        Humano humano = new Humano();
        System.out.println(humano.nome);
        humano.falar();
    }
}

class Animal{
    String nome;
    int idade;
    float peso;
    String cor;

    void falar(){
        System.out.println("Olá");
    }
    void andar(){
        System.out.println("Andou");
    }
}

class Humano extends Animal{

}
```

Parabéns querido aluno por chegar até aqui e ter adquirido mais este valioso conhecimento!

Se quiser aprender sempre mais sobre criação de Jogos e Aplicativos, não deixe de conhecer o Aprenda Programar, meu portal de cursos online onde você pode se especializar em:

- Algoritmos e Lógica de Programação
- Modelagem e Animação 3D
- Criação de Personagens para Jogos e Filmes
- Programação de Aplicativos Nativos para iOS e Android
- Criação de Games 2D, 3D e Realidade Virtual
- Realidade Aumentada e Visão Computacional
- Metodologia STEAM
- Robótica e Impressão 3D

Para virar aluno do Aprenda Programar você deve se inscrever pela plataforma Hotmart, no link abaixo.

Adquira seu acesso para sempre ao Aprenda Programar:
https://hotmart.com/product/en/aprenda-programar-com-tito-petri